Indian Health Service
Information Technology Support Center

RPMS Programming Standards and Conventions

January 2003

Table of Contents

Indian Health Service
Information Technology Support Center

Purpose, Policy, & The Standards and Conventions Committee

January 2003

1 PURPOSE, POLICY, & THE STANDARDS AND CONVENTIONS COMMITTEE

1.1 PURPOSE

The purpose of this document is to provide a cornerstone for all national software distributed throughout the Indian Health Service (IHS). Programming Standards and Conventions (SAC) mandate functional soundness and technical correctness of Resource & Patient Management System (RPMS) programs and all other programs that interface with RPMS. RPMS software includes programs written in M and other programming environments.

1.2 POLICY

1.2.1 Conformance

All RPMS software developed for IHS will conform to the Programming SAC. In areas where specific IHS standards have yet to be adopted, software development will conform to accepted industry standards.

1.2.2 Quality Control

The IHS Programming SAC have been established as a key aspect of quality control under the Software Engineering Team (SET), Information Technology Support Center (ITSC), and Division of Information Resources (DIR).

1.2.3 Definition/Appeal

The IHS Standards and Conventions Committee (SACC) defines Programming SAC and serves as the source of appeal when issues of conformity with Programming SAC arise.

1.2.4 Development

SACC members and other technical experts within IHS will participate in the development of Programming SAC in collaboration with public and private subject experts and with national and international standards organizations.

1.3 SACC CHARTER

1.3.1 Purpose

The Standards and Conventions Committee (SACC) was established to define and review national standards for software developed within the Indian Health Service

(IHS). This applies to SET efforts to modernize IHS systems, as well as to other ITSC development for RPMS and to area and local facility development. The SACC will be responsible for promoting consistency in standards development by authoring the Programming Standards and Conventions (SAC) and the Graphical User's Interface Standards and Conventions (GUI SAC). The SACC is also to serve as an arbiter when issues of interpretation to SAC arise.

a. Requests for exemptions from existing RPMS standards are made by the development community to the SACC via written or electronic media.
b. A simple majority decision by the committee will be final regarding interpretation or waivers of the standards.

1.3.2 Mission

The mission of the SACC is to:

a. Provide a cornerstone for all national software developed and distributed within Indian Health Service (IHS);
b. Provide mandates for functional soundness and technical correctness of RPMS and all other programs written in the M programming language as well as RPMS programs written for the desktop environment;
c. Provide a base of utilities and techniques for uniformity;
d. To rule on disputes about the interpretations of the SAC; and
e. To make recommendations on requests for SAC exemptions.

1.3.3 Responsibilities

a. The membership of the SACC shall include representatives from the Software Engineering Team (SET). The SACC may call upon individuals outside of its membership on an ad hoc basis as necessary to serve on focused work groups involved in SACC activities.
b. The SACC will report directly to the SET Chief. Decisions and/or recommendations will be forwarded to the SET Chief for approval.
c. The Chairperson of the SACC will forward a quarterly report detailing committee activities to the SET Chief.
d. A committee member, selected by the entire SACC and approved by the SET Chief, will chair the committee, organize agendas, and track action items.
e. SACC meetings will be held at least once a quarter via teleconference. These meetings shall be focused on updating the SAC document, processing requests for exemption, and issuing interpretations to clarify the SAC document. Other ad-hoc meetings to discuss specific assignments may be held as needed or at the request of the Chairperson or the SET Chief.
f. The SET Chief will sponsor the Standards and Conventions Committee, including approval of SACC recommendations.
g. SACC members and other technical experts within IHS will participate in the development of the Programming SAC in collaboration with public and private subject experts and with national and international standards organizations.
h. The SET Chief is responsible for promulgating all programming SAC to IHS programmers via IHS MailMan and other electronic media (e.g., web pages).
i. Each Software Development Project Team Lead is responsible for assuring that

programming SAC is followed.

1.3.4 Membership

The SACC membership consists of one representative from each of the following areas:

- Area ISC Representative
- Clinical Applications, SET
- Administrative Applications, SET
- Infrastructure Applications, SET
- Software Quality Assurance, SET

And 2 members at large for a total membership of seven voting members. ITSC members will be selected by their appropriate Team Leader.

a. An alternate member may be appointed for each SACC member. An alternate does not have voting privileges unless the alternate is acting on behalf of the full member.
b. Membership period is unrestricted and is at the discretion of the SET Chief in consultation with other ITSC Team Leaders.
c. The SACC will elect a chairperson to serve a 2-year term. The responsibilities of the chairperson role are to:
    - Coordinate the establishment of new standards as needed;
    - Coordinate standards revisions;
    - Coordinate waiver and/or exemption requests; and
    - Report committee decisions to the programming and ITSC management communities.

1.4 SACC PROCEDURES

1.4.1 Proposals for Changes

1.4.1.1 Proposals

If an individual identifies the need for a change in the SAC, a request must be made to the SACC either via written notification using the form in Appendix B or the SACC WebBoard. Proposals should be submitted as early as possible in the development cycle.

1.4.1.2 Proposal requests

Proposal requests should include the following information:

- Current section and wording
- Recommended change

- Justification and comments

1.4.1.3 SACC Review

After review of the need and impact of the proposed change, the Chairperson of the SACC will call for a vote on the request. A simple majority decision by the SACC shall be required to grant a change to the SAC.

1.4.1.4 Decision Notification

Notification of the decision of the SACC will be made to the individual requesting the change and will be forwarded to the Team Lead of ITSC, where it will be processed as a new programming SAC.

1.4.2 New Programming SAC

1.4.2.1 Publication of Changes

The SACC will publish the proposed Programming SAC changes (either individual paragraph proposals or the entire document) onto the SACC WebBoard for comments at least 30 days prior to voting on the proposed changes.

1.4.2.2 Approval

After the SACC votes to accept the proposed Programming SAC, the SACC will forward it to the SET Chief for approval. If the SET Chief does not respond with approval/disapproval within 30 days from the day it is submitted to him/her, the SAC is considered approved. If the SET Chief approves the SAC document within the 30 days, it is considered effective immediately. If the SET Chief disapproves the SAC document, the SACC will make the appropriate changes and resubmit it to the SET Chief, repeating the process until the SET Chief approves the SAC document as indicated above.

1.4.2.3 Effective Date

The new SAC becomes effective on the date of approval by the SET Chief. After approval, the new SAC will be announced and made available on the SACC WebBoard.

1.4.3 Conformance

Following the effective date of the new Programming SAC, newly submitted RPMS packages must adhere to the new SAC unless otherwise exempted as presented in this document.

1.4.4 Requests for Exemptions from Programming SAC

1.4.4.1 Procedure

If an individual identifies the need for an exemption from the SAC for a given package, a request must be made to the SACC either via written notification using the form in Appendix B or the SACC WebBoard. Exemption requests should be submitted as early as possible in the development cycle.

1.4.4.2 Requirements

Exemption requests should include the following information:

- Package name and version number
- Type (Permanent or temporary)
- Violated Standard
- Justification and comments

1.4.4.3 SACC Review

After reviewing the need and impact of the proposed exemption, the Chairperson of the SACC will call for a vote on the request. A simple majority decision by the SACC shall be required to recommend a SAC exemption to the SET Chief.

1.4.4.4 Decision Notification

Notification of the decision of the SET Chief will be made to the requestor and will be published on the WebBoard.

1.4.5 Publication of Current Programming SAC

In order to promote rapid dissemination, the current version of the IHS SAC will be published on the SACC WebBoard for all users to access. All developers should recognize the need for periodic review of the SACC WebBoard.

Indian Health Service
Information Technology Support Center

Programming Standards and Conventions

January 2003

2 RPMS PROGRAMMING STANDARDS AND CONVENTIONS (SAC)

This version of the Programming SAC was adopted by IHS on January 14, 2003.

2.1 GENERAL PROGRAMMING STANDARDS AND CONVENTIONS

The definitions, standards and conventions within this section apply to all programming and user environments for use in the RPMS. This includes all applications that use commercial software products as an integral part of the RPMS (but not to the commercial application itself) and to all development environments including, but not limited to programming languages such as M or Pascal and graphical user interface (GUI) environments such as Visual Basic, Delphi, etc.

2.1.1 Document Definitions

2.1.1.1 Application Programmer Interface (API)

See Published Entry Point (PEP)

2.1.1.2 Conventions

Programming guidelines that are designed to promote consistency and safety across RPMS applications. Exemptions from conventions are not required, but developers are strongly encouraged to follow them.

2.1.1.3 DBA

Database Administrator

2.1.1.4 DBIC

Database Integration Committee

2.1.1.5 DHCP

Decentralized Hospital Computer Program. Department of Veterans Affairs (VA) software equivalent to the RPMS, now called VistA.

2.1.1.6 Entry Point

A line label within a routine, other than the first line of the routine, that is referenced by a "DO" or "GOTO" command from another routine in the same package to which the routine belongs.

2.1.1.7 Exemption

Authority granted by the SACC to a specific RPMS/VistA application that allows that application to not comply with a particular section of the SAC for a specified timeframe.

2.1.1.8 Kernel

The Kernel is a set of VA software utilities that form the foundation of VistA/RPMS and include elements that start with the namespaces XG*, XLF*, XPD*, XQ*, XT*, XU*, XV*, ZI*, ZO*, ZT*, ZU*.

2.1.1.9 MailMan

MailMan is a set of VA software utilities that form the foundation of VistA/RPMS electronic mail and communications and include elements that start with the namespace XM*.

2.1.1.10 Namespace

A unique set of alpha characters assigned by the DBA, commonly used to uniquely identify package components.

2.1.1.11 Numberspace

A unique set of numbers assigned by the DBA, commonly used for assigning FileMan files.

2.1.1.12 Package

A set of routines, files, options, templates, security keys, screens, bulletins, functions, help frames, forms, blocks, objects, protocols, dialogues, list templates, windows, etc., namespaced according to DBA requirements that function as a unit.

2.1.1.13 Programmer Mail Group

The Programmer mail group (G. Programmer) is established on the IHS MailMan system for discussion of technical issues. This mail group is used to disseminate information to the IHS RPMS development community. (Subsequent changes to the SAC will indicate a migration from the use of MailMan to the use of WebBoard.)

2.1.1.14 Published Entry Point (PEP)

A line label in a routine, other than the first line of a routine, which has been internally documented as a published entry point, and documented in the technical manual as a published entry point, that is available to be referenced by a "DO" or "GOTO" command from a routine external to the package to which the routine belongs. (This is also known as an Application Programmer Interface (API).)

2.1.1.15 SACC WebBoard

A documentation and conversation facilitator based on Internet technology for issues related to the IHS Standards and Conventions, found at http://forum.ihs.gov/~sacc.

2.1.1.16 Standard

A rule that all RPMS/VistA software must follow.

2.1.1.17 Supported Reference

Routine labels, extrinsic functions, files or global nodes that are accepted and documented by the DBIC and listed on the DBA menu on IHS Mail.

2.1.1.18 User

A person interacting with computer applications.

2.1.1.19 Utility

A callable routine line tag or function. A universal routine like XB*, ZIB*, AUPN*, usable by anyone.

2.1.1.20 Up-hat

"^" which is denoted on the keyboard by pressing Shift+6, a circumflex, also known as "hat" or "caret". It is used as a piece delimiter in a global.

2.1.1.21 VA FileMan

The Database Management System for RPMS/VistA, with namespaces DD*, DI*, and DM*.

2.1.1.22 VistA

Department of Veterans Affairs (VA) software equivalent to the RPMS

2.1.2 Files

2.1.2.1 Naming Requirements for Files Used by RPMS/VistA Packages

2.1.2.1.1 VA FileMan

All VA FileMan files in the M Language environment must be numberspaced and namespaced in the spaces assigned to the package by the DBA. See Appendix A.

2.1.2.1.2 DOS, VMS, UNIX or Other Host Files

All DOS, VMS, UNIX or other host files created or exported as part of a RPMS/VistA application must be namespaced in the namespace assigned by the DBA. See Appendix C.

2.1.2.2 Exporting Script Files

Packages exporting script files should provide script files for a variety of the terminal emulation packages commonly in use in the VA/IHS.

2.1.2.3 Exporting Spreadsheets

Packages exporting spreadsheet templates should apply protection to embedded formulas to prevent accidental deletion by a user. Spreadsheet templates should contain documentation describing the purpose of the template, complex functions, and user help.

2.2 M LANGUAGE PROGRAMMING STANDARDS AND CONVENTIONS

All M-based RPMS/VistA software will meet the following standards and comply with the spirit of the conventions.

2.2.1 ANSI Standards

The 1995 ANSI/MDC X11.1 Sections 1 and 2 will be adhered to unless explicitly modified by this document.

2.2.1.1 Standard Dictionary Releases

All development will be produced using the most recent standard dictionary releases as distributed by the ITSC.

2.2.1.2 Fields Within A String

Fields within a string will be separated by the "up-hat" symbol except when the data within the string must contain the "up-hat" itself.

2.2.1.3 Reindexing of File Manager MUMPS Cross-References

Reindexing of individual File Manager MUMPS cross-references must not result in an error. This affects both "set" and "kill" logic.

e.g., Use I $D(XYZ),XYZ'="" S ^GBL("AC",XYZ)=""

2.2.1.4 FileMan Entry Points

Only standard documented FileMan entry points will be called by IHS routines.

2.2.1.5 Restrictions When Creating a Package For Distribution

All packages submitted for certification for release to I/T/Us will be in a KIDS build, will contain only those components that belong to the package, and will not contain global data except as allowed by KIDS. Components that are part of one package will not normally be distributed with another package. Exceptions are existing authorized applications that are related but require separate KIDS builds, which can be placed in a composite transport.

2.2.1.6 File Manager File Access Code Security

All packages delivered to the SRCT for verification and distribution will come with complete File Manager file access code security in the data dictionary. All files will have programmer only data dictionary access. This assures that security will be present initially if dictionaries are deleted prior to installation. The only exception to having file access security codes is in the READ field in a DD in which case it can be without access or blank. If developers need file access codes for their packages, they will coordinate this with the DBA.

2.2.1.7 Version Information

Complete version information will be set into the package file (DIC(9.4)) and all files for a package will have the version number in DD(file#,"VR").

2.2.1.8 LAYGO Restriction

Application programs will not allow LAYGO entry to the New Person file. Packages which need to be able to add to this file will have separate locked options that come with the package, but that are not assigned to any of the package menus as distributed. Application programs will not allow "LAYGO" to standard tables.

2.2.1.9 Entry in File 9000010

Packages external to PCC that are passing data to the Visit and V-files must use the APCDALV routine to select or create a visit entry in file 9000010. The APCDALVR will be used to append a V-file entry to a visit file entry.

2.2.2 Routines (Routine Structure and Format)

2.2.2.1 First Line

The first line of a routine will have the following:

(routine name) ; (Agency)/(site)/(programmer)-(brief description) ; (date of edit)

e.g., AGPAT ;IHS/OKC/LD - Patient registration driver; JUL 20, 1986

2.2.2.1.1 First Line

The first line of a routine cannot contain the formal list for parameter passing.

2.2.2.1.2 Generated Routines

Routines generated by VA FileMan or Kernel (e.g., INITs, ONITs, NTEG, and compiled routines) and other compiled routines used in exporting a package, need not comply with this standard.

2.2.2.1.3 Agency/Site/Programmer

This shall identify the developing Agency and site, and the programmer who is currently responsible for maintenance and development of the package.

2.2.2.1.4 Date of Edit

The use of time in date of edit is optional.

2.2.2.2 Second Line

The second line of a routine must be in the following format:

LABEL-optional<ls>;;version number;package name;**pm,...pn**;version date

Where:

2.2.2.2.1 Version Number

The version number must be the same on all of the package-namespaced routines including the Inits, Onits, etc.

2.2.2.2.2 Patch Numbers

"pm,...pn" are the applied patch numbers separated by commas, this ";" piece is null if there are no patches.

2.2.2.2.3 Version Date

The version date must be the same on all of the package namespaced routines including the Inits, Onits, etc.

2.2.2.2.4 Compiled routines

Routines compiled from templates, cross-referenced, etc., by VA FileMan during or after package installation are exempt from the second line requirement.

2.2.2.2.5 Local Modifications

If local modifications to a routine are restricted or prohibited by policy or directive, the third line should contain an appropriate notice. (e.g., "Per VHA Directive 10-92-142, this routine should not be modified")

2.2.2.2.6 Labels

Labels are limited to eight (8) characters and may not contain lower case characters.

2.2.2.2.7 Label+Offset

Label+offset references will not be used except for $TEXT references.

2.2.2.2.8 Lines Referenced by $TEXT

Lines referenced by $TEXT for use other than to check for the existence of the routine must be in the following format: (LABEL-optional)<ls>;;text or M code.

2.2.2.3 Linebody

The linebody must contain at least 1 character, must not exceed 245 characters in length, and must contain only the ASCII characters values 32-126. Commands,

functions, local and global variable names, system variables, SSVNs, etc., must be uppercase.

2.2.2.4 Routine Names

Package routine names of the following forms will not be used:

- **NAMESPACE_I*** (with the exceptions of Kernel, VA FileMan, and routines created to support the INIT process)
- **NAMESPACE_NTE*** (with the exception of the package integrity routines)

2.2.2.5 Routine Size

The maximum routine size, as determined by executing ^%ZOSF("SIZE"), is 15,000 characters. The combination of routine and symbol table must run in the partition size specified in the appropriate VA/RPMS operating system manual/cookbook.

2.2.2.6 Vendor Specific Subroutines

Vendor specific subroutines may not be called directly except by Kernel, MailMan and VA FileMan.

2.2.2.7 TaskMan Globals

All applications will use documented TaskMan utilities to interface with TaskMan globals.

2.2.2.8 Naked References

Naked references must either be appropriately preceded by the full reference defining it or be documented.

2.2.2.8.1 Full Reference

An appropriate preceding full reference is one that is on the same physical routine line as the naked reference and has no code between it and the naked reference that branches in any manner to other lines of code or executables.

2.2.2.8.2 Documenting

Those naked references requiring documentation must be documented within the routine in the immediate vicinity of the naked reference. Those naked references that are preceded by a full reference that is outside of the routine where the naked reference is used must have documentation in both the routine containing the full reference and the routine containing the naked reference. This documentation must be in the immediate vicinity of the appropriate reference.

2.2.2.8.3 In Called Utilities

Uses of naked references in called utilities are exempt, e.g., S DIC=200,DIC(0)="AEQ",DIC("S")="I $L($P($G(^(1)),"^",9))" D ^DIC is a legitimate use of the naked reference.

2.2.2.9 % Routines

2.2.2.9.1 % Routines

No application will distribute % routines. (Exemptions: Kernel and VA FileMan)**.** No % routines shall execute variables that could be set by a programmer prior to executing the code.

2.2.2.9.2 View Commands

No routine that will be resident in the Library (MGR) account will use VIEW commands using variables as arguments that could be set by a programmer prior to executing the code.

2.2.2.10 Z Routines

2.2.2.10.1 Z Routines

No application will export routines whose names start with the letter "Z". (Exemption: Kernel)

2.2.2.10.2 Creation of Local Routines

When creating local routines to be added to an existing national package, the routine name will begin with the namespace followed by the letter "Z".

2.2.2.11 Extended Reference Syntax

Routines may not be invoked using the extended reference syntax, i.e., D ^|"VAH"|TAG^ROUTINE is illegal.

2.2.2.12 Entry Points

Lines that are entry points (referenced by a DO or GOTO in other routines) will consist only of a label and a semi-colon followed by "entry point" or the abbreviation of "EP". An optional comment that describes the entry point may follow the entry point comment.

e.g., CLEAR ; EP - CLEARS SCREEN

e.g., CLEAR ; ENTRY POINT - CLEARS SCREEN

2.2.2.13 Published Entry Points

Lines in a routine, other than the first line, that are published entry points (PEP) must consist of a label (and, optionally, the formal parameter list) and a semicolon

followed by “Published Entry Point” or the abbreviation “PEP,” followed by a comment.

2.2.2.14 Changed Lines

Lines changed from the standard release will be marked with a semicolon and comment. This comment will contain the Agency and site identification, the programmer's initials, the date of the change, and reason for the change. Duplicate and comment out the original line of code. Additional comment lines may be used.

```
e.g.,     CHGLINE;
 ;S ZXXX=3
        S AXXX=3 ; IHS/ABQD/FBD 04/01/96
;This was changed to conform to IHS SAC.
```

2.2.2.15 IHS Changes to VA Packages

IHS changes to VA packages will be incorporated into separate routines or templates using a proper IHS namespace, rather than embedding the changes in the middle of VA programs. (A "DO" statement will be used to exit to the IHS routines, if needed). When it is not possible to use an external IHS routine when modifying VA packages,add changes in the format outlined in Standard 2.2.2.14.

2.2.2.16 Comments

Comments will be provided at the start of each independently callable routine specifying the items listed in paragraphs 2.2.2.16.1 through 2.2.2.16.3.

2.2.2.16.1 Purpose or Function

Overall functions or purpose of routine.

2.2.2.16.2 Input Variables

Input variables (variables set up by other routines that are used by this routine).

2.2.2.16.3 Output Variables

Output variables (variables set by this routine).

2.2.2.17 XB/ZIB Prefixed Routines

AU/AZ prefixed utility routines have been re-namespaced XB/ZIB respectively. All XB prefixed utility routines will refer to utility routines that are totally universal from machine to machine regardless of operating system or hardware. All ZIB prefixed utility routines will refer to utility programs that are operating system or hardware dependent. All program calls to previous AU/AZ prefixed utility routines must be changed to make the program calls to the XB/ZIB equivalent utility routines.

2.2.2.18 Facility or Area Specific Information

Facility or Area specific information needed by a routine will be obtained from tables or input parameters, rather than being hard-coded in the routine.

2.2.2.19 Approved Exemptions

Approved exemptions to standards must be documented in the routine with a separate comment line adjacent to the line containing the exemption in the form:

;IHS exemption approved on DATE

2.2.2.20 "Namespace" Var Routine

Packages that need to set local variables prior to entering a package will use a routine that is named in the form 'namespace' VAR.

2.2.2.21 Data Conversion Processes

All data conversion processes should be restartable at the point of interruption in the execution.

2.2.2.22 Syntactically Correct Lines

All lines must be syntactically correct. Blanks will not appear at the end of lines.

2.2.3 Variables

2.2.3.1 Local Variables

2.2.3.1.1 Case

Lowercase characters in local variable names are prohibited.

2.2.3.1.2 Length of Local Variables

The full evaluated length of a local variable name including subscripts must not exceed 200 characters. The evaluated length is calculated as follows

Example subscripted variable:

NAME(sub1,sub2,...,subn)

(1.) +$L(NAME)+3

(2.) +$L(sub1) + $L(sub2) + ... +$L(subn)

(3.) + 2 * number of subscripts n

(4.) +15

VAR("XXX",123,1,2,0) would evaluate to a string length of 42

(6+11+10+15)=42.

2.2.3.1.3 System-Wide Variables

2.2.3.1.3.1 Variables

The following are system wide variables. Any application setting system-wide variables must conform to the following definitions.

- **AGE** - Patient age in years from date of birth to DT expressed as an integer, or, if deceased, the date of death
- **DFN** - Internal number of an entry in the Patient File (#2)
- **DOB** - Patient date of birth expressed in internal VA FileMan format
- **SEX** - Patient sex; either "F" or "M"
- **SSN** - Social security number with 9 contiguous digits, or 9 digits and a "P"
- **VA ("BID")** - Brief patient identifier; up to 7 characters
- **VA ("PID")** - Patient identifier; up to 15 characters

2.2.3.1.3.2 Assumed Variables

The DT*, DUZ*, IO*, and U variables, referenced elsewhere in this document, are set by Kernel during sign-on, or by VA FileMan, and can be assumed to exist by all VistA/RPMS applications. Refer to the Kernel Systems Manual or the VA FileMan Technical Manual for their definitions.

2.2.3.1.4 DUZ or DUZ-Array

RPMS/VistA packages are not allowed to KILL, NEW, SET, MERGE, READ (into) or otherwise modify the variable DUZ or any DUZ-array element with the exception of DUZ(2) (Exemptions: Kernel and VA FileMan). In order to allow programs to run stand alone or for transport to non-Kernel environments, DUZ and its descendants can be set after confirming that they do not already exist.

2.2.3.1.5 DUZ(2)

The VA local variable DUZ(2) will be set for all users upon logon by Kernel, using the FACILITY field in the New Person. A user will be able to log on to only those sites listed under the FACILITY field in his or her New Person file entry. Any application allowing a user to switch facilities in a package will use this multiple field in the New Person file when selecting a valid site for the user to switch to. If a package modifies DUZ(2) after the original set, it will be reset to its original value before exiting the package. A value of 0 in DUZ(2) or the existence of the local variable AUPNLK("ALL") will allow complete lookup ability for all facilities in the Patient File.

2.2.3.1.6 Special Variables

The variables DT, DTIME, and U have no array elements and shall be initially defined by Kernel or VA FileMan.

2.2.3.1.6.1 Variable U

The variable U will not be KILLed or NEWed or changed from the value defined by Kernel or VA FileMan. (It is legal to SET U="^".)

2.2.3.1.6.2 Variable DT

The variable DT will not be KILLed or NEWed. If changed it must be set using the supported reference S DT=$$DT^XLFDT.

2.2.3.1.6.3 Variable DTIME

The variable DTIME may only be changed to a value less than the existing value of DTIME, but must be restored to its original value before exiting the option. All routines where DTIME is changed but not restored prior to exiting the routine must be documented in the Technical Manual, including the location where DTIME is restored.

2.2.3.1.7 IO Variables

RPMS/VistA packages are not allowed to KILL, SET, MERGE, READ (into) or otherwise modify IO namespaced variables and any of their array elements except those documented as modifiable in the Kernel System Manual. (Exemption: Kernel, MailMan, and VA FileMan) The routine XBKVAR can be invoked to set these variables.

2.2.3.1.8 % Variables

RPMS/VistA packages are not allowed to KILL, SET, MERGE, READ (into) or otherwise modify % variables. Exceptions to this are the single character variable "%" and the variables set for and/or returned by Kernel and VA FileMan supported references. (Exemption: Kernel, VA FileMan, and MailMan)

2.2.3.1.9 Scope of Namespaced Variables

A RPMS/VistA package may declare namespaced, local variables as package-wide. The variables and all of their array elements must be described in the package Technical Manual. A RPMS/VistA package may not kill or change another RPMS/VistA package's package-wide variables.

2.2.3.1.10 Documentation

Documentation on how to create and kill package-wide variables created by an option that is removed from its exported menu path must be included in the Technical Manual.

2.2.3.1.11 Lock Tables/Local Symbol Tables

All supported references must leave the lock tables and local symbol tables unchanged upon exit with the exception of the following:

- Documented input and output variables (including globals)
- Supported reference namespaced variables may be changed or killed (for example, the VA FileMan ^DIC call kills the variable DIE, which may exist in the symbol table prior to the call)
- Documented side effects, such as lock table changes, and changes to files
- The variable %

These supported references must be documented in the package Technical Manual with a descriptive list of ALL input and resulting output variables.

2.2.3.1.12 Variables Passed Between Packages

Naming requirements for variables passed between packages.

2.2.3.1.12.1 Actual List

Input variables in an Actual List passed by reference between packages must be package namespaced.

Legal:

```
D BLD^DIALOG(3500010,"",.IBDATA,"IBX")
```

Illegal: D BLD^DIALOG(3500010, "",.Y,"IBX")

2.2.3.1.12.2 Input Variables

Input variables that represent local variables into which data will be exchanged must represent a data location that is package namespaced.

Legal:

```
S DA=10,DR=".01;.104",
DIC="^DPT(",DIQ="IBX" D EN^DIQ1
```

Illegal:

```
S DA=10,DR=".01;.104",
DIC="^DPT(",  DIQ="Y"  D EN^DIQ1
```

2.2.3.2 Global Variables

2.2.3.2.1 Global Name

Lowercase characters in global names and global subscripts are prohibited. (Exemption: Cross-references created using field values containing lowercase characters and subscripts used in the ^TMP and ^XTMP globals.)

2.2.3.2.2 Length of Global Reference

The full evaluated length of a global reference must not exceed 511 characters. The evaluated length is calculated as follows.

Example subscripted variable:

^NAME(sub1,sub2,...,subn)

(1.) +$L(NAME)+3

(2.) +$L(sub1) + $L(sub2) + ... +$L(subn)

(3.) + 2 * number of subscripts n

(4.) +15

^TMP("XXX",123,1,2,0) would evaluate to a string length of 42

(6+11+10+15)=42.

2.2.3.2.3 Unsubscripted Globals

The KILLing of unsubscripted globals is prohibited. (VA FileMan's EN^DIU2 utility allows the deletion of files stored in unsubscripted globals, and therefore, allows the killing of unsubscripted globals. Application developers must document when calls to EN^DIU2 are made to delete files stored in unsubscripted globals.)

2.2.3.2.4 %Globals

READing, KILLing, SETting or MERGing ^% globals is prohibited. (Exemption: Kernel) %Globals will not be created without approval from the SET.

2.2.3.2.5 Globals

All globals must be VA FileMan compatible. ^TMP, ^XTMP and ^UTILITY have a standing exemption from this requirement.

2.2.3.2.5.1 ^TMP Global

The global ^TMP will be used as a scratch global within a session. The first subscript shall be $J, or the first two subscripts shall be a package namespaced subscript followed by $J. The ^UTILITY global will not be used unless necessary to retrieve output from a Kernel or FileMan Utility.

2.2.3.2.5.2 ^XTMP Global

The global ^XTMP will be translated, with one copy for the entire RPMS production system at each site. The structure of each top node shall follow the format ^XTMP(namespaced- subscript,0)=purge date^create date^optional descriptive information, and both dates will be in VA FileMan internal date format.

2.2.3.2.6 Executable Code

Fields in VA FileMan files that contain executable code must be write protected in the DD with "@" (e.g., ^DD(file,field,9)="@"), or be defined as VA FileMan data type of "MUMPS".

2.2.3.2.7 DD Global

Sets and Kills to the DD Global are prohibited.

2.2.3.2.8 Global Variables

All global variables executed by % routines must be in write protected globals.

2.2.3.2.9 Global Names

Extended reference syntax may not be used to reference global variables, i.e.,

S X=^|"VAH"|GLOBAL(1,1) is illegal.

2.2.3.3 Intrinsic (System) Variables

2.2.3.3.1 Intrinsic Variables

Lowercase intrinsic variables are prohibited.

2.2.3.3.2 Intrinsic Variables

No VistA/RPMS package may use the following intrinsic (system) variables unless they are accessed using Kernel or VA FileMan supported references: $D[EVICE], $EC[ODE], $ES[TACK], $ET[RAP], $I[O], $K[EY], $P[RINCIPAL], $Q[UIT], $ST[ACK], $SY[STEM], $Z*. (Exemption: Kernel and VA FileMan)

2.2.3.4 Structured System Variables (SSVNS)

2.2.3.4.1 SSVNS

Lowercase SSVNs are prohibited.

2.2.3.4.2 Restricted SSVNS

The following structured system variables may be used only by Kernel or VA FileMan or through their supported references: ^$CHARACTER, ^$DEVICE, ^$DISPLAY, ^$EVENT, ^$GLOBAL, ^$JOB, ^$LOCK, ^$ROUTINE, ^$SYSTEM, ^$Z*, and ^$WINDOW.

2.2.4 Commands

2.2.4.1 Case

Lowercase commands are prohibited.

2.2.4.2 BREAK Command

Direct use of the BREAK command is prohibited. Use ^%ZOSF("BRK") and ^%ZOSF("NBRK"). (Exemptions: Kernel and VA FileMan)

2.2.4.3 CLOSE Command

Direct use of the CLOSE command is prohibited. Use the routine ^%ZISC. (Exemptions: Kernel, MailMan and VA FileMan)

2.2.4.4 HALT Command

Direct use of the HALT command is prohibited. Use the supported reference H^XUS. (Exemption: Kernel and VA FileMan)

2.2.4.5 JOB Command

Direct use of the JOB command is prohibited. Use the Kernel Task Manager's supported calls to create jobs. (Exemption: Kernel and MailMan)

2.2.4.6 KILL Command

2.2.4.6.1 Argumentless

The argumentless form of the KILL command is prohibited. (Exemption: Kernel)

2.2.4.6.2 Exclusive

The exclusive form of the KILL command is prohibited. (Exemptions: Kernel and VA FileMan)

2.2.4.7 LOCK Command

2.2.4.7.1 LOCK

All LOCKs shall be of the incremental or decremental form. All routines will lock nodes to be created or updated. Appropriate error recovery action will be taken if the lock is not obtained. (Exemption: Kernel)

2.2.4.7.2 Incremental LOCK

All incremental LOCKS must have a timeout.

2.2.4.8 NEW Command

2.2.4.8.1 Argumentless NEW

The argumentless form of the NEW command is prohibited.

2.2.4.8.2 Exclusive NEW

The exclusive form of the NEW command is prohibited. (VA Only)

2.2.4.9 OPEN Command

The use of the OPEN command is prohibited. (Exemptions: Kernel, MailMan and VA FileMan)

2.2.4.10 READ Command

2.2.4.10.1 READ

All READ commands shall read into local variables, or one of the non-VA FileMan compatible globals, ^TMP and ^XTMP.

2.2.4.10.2 READ

All user input READs must have a timeout. If the duration of the timeout is not specified by the variable DTIME and the duration exceeds 300 seconds, documentation in the package Technical Manual is required.

2.2.4.10.3 READ

All user input READ commands shall be terminated by a carriage return. (Exemption: Kernel and VA FileMan) (Developers desiring to implement escape processing [function keys, arrow keys, etc.] must use Kernel supplied supported references [XGF]).

2.2.4.10.4 READ

Use of the READ command in a data dictionary is prohibited.

2.2.4.11 USE Command

The use of the USE command with parameters is prohibited. (Exemptions: Kernel and VA FileMan)

2.2.4.12 VIEW Command

The use of the VIEW command is prohibited. (Exemptions: Kernel and VA FileMan)

2.2.4.13 MWAPI Commands

No RPMS/VistA package may use the MWAPI Commands, ESTART, ESTOP, and ETRIGGER. (Exemption: Kernel)

2.2.4.14 WRITE

Use of the WRITE command in a data dictionary is prohibited. The call to EN^DDIOL will be used (Exemption: VA FileMan).

2.2.5 Z* Commands

The use of Z* commands is prohibited. (Exemptions: Kernel and VA FileMan)

2.2.6 Functions

2.2.6.1 Case

Lowercase functions are prohibited

2.2.6.2 Intrinsic Functions

2.2.6.2.1 $NEXT

Use of the $NEXT function is prohibited. All RPMS packages should remove all occurrences of $NEXT. This includes occurrences generated by VA FileMan.

2.2.6.2.2 $VIEW

The use of the $VIEW function is prohibited. (Exemptions: Kernel and VA FileMan)

2.2.6.2.3 $Z*

The use of $Z* functions are prohibited. (Exemptions: Kernel and VA FileMan)

2.2.6.2.4 $ORDER

Reverse $ORDER through the local symbol table when the variables are un-subscripted is not supported by Cache' and is prohibited. Reverse $ORDER through subscripted local variables is supported and allowed.

2.2.6.3 Extrinsic Functions

2.2.6.3.1 Parameters

Supported References that use parameters will document the elements of the formal list internally within the routine and in the package Technical Manual. Documentation will specify which elements of the formal list are required and which are optional, if any, and those elements that must be passed by reference.

2.2.6.3.2 Supported Extrinsic Special Variables

Extrinsic functions with an empty formal list will be documented within the routine and in the Technical Manual.

2.2.7 Name Requirements

2.2.7.1 Package Namespace

Unless approved by the DBA, routine, global, security key, option, template, bulletin, function, screen, help frame, protocol, form, block, list templates, objects, dialogues, remote procedures, etc., names must be consistent with the

assigned DBA namespace for the package. The namespace of all IHS-developed packages assigned after January 1, 1994 must begin with *B*.

2.2.7.2 "Namespace"Menu

All IHS packages will have a top menu option in the form of "namespace"MENU with a lock on that menu option that is of the form 'namespace'ZMENU. When entering this menu the current version of the package will be displayed above the options along with the location determined by the current DUZ(2) setting.

2.2.8 Options (Option file entries)

2.2.8.1 Selection

Option selection must be made through Menu Manager or using VA FileMan's DIR utility. DIR utility may only be used at the lowest menu tree level. Hardcoded menu management systems are not allowed.

2.2.8.2 Path Independence

All options in a package must be path independent once the steps described in the Technical Manual for creating and killing package wide variables have been taken.

2.2.8.3 After Exit

The following must not exist after exiting an option:

2.2.8.3.1 Output Variables

All documented output variables created by a called supported reference.

2.2.8.3.2 Locks

All documented locks created by a called supported reference.

2.2.8.3.3 Global Nodes

All documented temporary scratch global nodes (e.g., ^TMP and ^UTILITY) created by a called supported reference, with the exception of ^XTMP global data.

2.2.8.3.4 Variables/Locks/Globals

All local variables, locks, and scratch global nodes (except ^XTMP, or other scratch globals designed to be passed between parts of a package) created by the application.

2.2.8.4 Menu Options

All menu options in a package will utilize alpha synonyms, not numeric. This is to allow use of the Kernel menu jumping capability.

2.2.9 Device Handling

2.2.9.1 Device Handling

All device selection and closing will be made through the use of the Kernel supported references. See Sections 6.3 and 6.9 for specific information about the OPEN and CLOSE Commands.

2.2.9.2 Output Queuing

All output to a hard copy device (e.g., printer) must allow for queuing or use of an auxport printer. Internal device selection must be by device name rather than by $I. Developers should bear in mind that either IO or $I variables may be non-numeric. Forced queuing, which is sometimes desirable, will be local site parameter driven.

2.2.9.3 Outputs

Any output directed to a hard copy device (e.g., printer) will not start with a form feed or line feeds with the purpose of creating a form feed, and will leave the device at top-of-form when the output is finished.

2.2.10 Date Processing

2.2.10.1 Manipulation

Any date processing will accommodate the century.

2.2.10.2 Date Display

Date display will include the century where there is a potential for ambiguity.

2.2.11 Type A Extensions

No Type A extensions to the M Language standard are currently approved for use.

2.2.12 Miscellaneous Standards

2.2.12.1 Implementation Specific Functions

Application software must use documented Kernel-supported references to perform all M operating system specific functions. (Exemptions: Kernel and VA FileMan) In addition, %ZISH is to be used for Host File functions.

2.2.12.1.1 VA FileMan

Application software must use documented VA FileMan standards such as defaults, editing text, date/time format, spacebar recall, help prompts, deleting stored values, control of logic flow, escapes, etc. (See VA FileMan Users Guide). Developers are encouraged to use documented FileMan calls to the fullest extent possible. (See VA FileMan Programmers Guide.)

2.2.12.2 Data Element Interpreted as Number

Any data element that may be interpreted as a number must contain no more than 15 significant digits.

2.2.12.3 Published Entry Points (PEP or API)

2.2.12.3.1 SET or KILL

A SET or KILL to another application's data must be done through an API supplied by that application. All other retrieval of information should be done through an API or standard FileMan calls.

2.2.12.3.2 IO

If the published entry point has IO, the PEP will have a parameter to allow silent (background) processing.

2.2.12.3.3 Phase Out

Packages may phase out supported references (as callable from outside the application and documented in the technical manual) by providing a minimum 18-month notice to the PROGRAMMER and ISC mail groups on IHS Mail.

2.2.12.3.4 Utilization

No application will make a call to another application except through documented published entry points or supported FileMan/Kernel calls.

2.2.12.4 Character Set

All globals and routines will use only the M character set profile.

2.2.13 Conventions

2.2.13.1 ^UTILITY

Only Kernel and VA FileMan and existing Supported References should use ^UTILITY.

2.2.13.2 Deleting Tasks

Tasks should be deleted from Task Manager's list by setting the variable ZTREQ equal to "@" just prior to the application QUITing.

2.2.13.3 Routine Documentation

2.2.13.3.1 Entry Points

Routine line tags referenced from outside the routine should be documented before, on or after the line tag. Documentation should include a description of function.

2.2.13.3.2 Supported References/Routines

All supported references or routines invoked initially from an option or protocol should contain documentation explaining the functionality and any required, passed input and output variables.

2.2.13.4 Device a CRT

The proper method of determining if a device is a CRT is to check that the variable IOST starts with the string "C-". (e.g., I $E(IOST,1,2)="C-")

2.2.13.5 ^XTMP

Developers are encouraged to include other descriptive information on the third piece of the 0 node of the XTMP global, such as task description and creator DUZ.

2.2.13.6 Location Name

All packages will display the location name determined by the current DUZ(2) setting above all menus of the package.

2.3 INTERFACE PROGRAMMING STANDARDS AND CONVENTIONS

It is the intention of this section of the SAC to provide a consistent path for users as applications migrate from scrolling mode to a screen mode (either ScreenMan, List Manager, or screen-oriented editors) to a GUI environment.

2.3.1 User Interface Standards for Scroll Mode and Screen Mode

2.3.1.1 Deletion

Deletion of a data value, if permissible, must be initiated by the user entering the at-sign "@".

2.3.1.2 READ

Escapes from user-input READs, if permissible, must be initiated by the user entering a circumflex "^".

2.3.1.3 Help

All prompts requesting user input must provide additional help when the user enters a question mark ("?"). Any unrecognized or inappropriate response must be handled properly; i.e., at a minimum in a manner similar to the way VA FileMan handles responses (see VA FileMan User's Manual). Responses to READs that are in no way evaluated by the application are excluded from this requirement.

2.3.1.4 Defaults

In scrolling mode, defaults must be so indicated with a double slash ("//") or "replace" indicating that "replace/with" editing is allowed. The null response (i.e., typing only the RETURN key) shall select the default.

2.3.1.5 READ Timeouts

When a user input READ command times out, if the argument of the read is in any way evaluated by the application, the program must return to the Menu Manager with no more than one intervening read. A timeout at the menu level must halt through H^XUS.

2.3.2 User Interface Conventions For Scroll Mode And Screen Mode

2.3.2.1 Terminology

Developers are encouraged to use the following terminology.

2.3.2.1.1 EXIT

Exit ends a function or application and removes from the screen all windows and pop-ups associated with that function or application. If information has been changed, the application may automatically save the information, or prompt the user to save or discard the information.

2.3.2.1.2 QUIT

Like Exit, Quit ends a function or application and removes from the screen all windows and pop-ups associated with that function or application. If information has been changed, the application may automatically discard the information, or prompt the user to save or discard the information.

2.3.2.1.3 CANCEL

Cancel allows users to back out of a function or application, one pop-up at a time, until they reach the highest-level window. At that point, another Cancel request has the same effect as an Exit action.

When users Cancel a pop-up, the application can decide whether to discard or retain the information in that pop-up, depending on how the application wants to establish the default values the next time the pop-up is displayed. If the information is discarded and the pop-up is later re-displayed, the pop-up contains the default values set by the application. If the information is retained and the pop-up is later re-displayed, the pop-up contains the same values as it did when the user canceled the pop-up.

2.3.2.1.4 CLOSE

Synonymous with Cancel.

2.3.2.2 Key Assignments

Developers are encouraged to use the following key assignments:

2.3.2.2.1 PF1 Key Gold

May result in different actions based on the next key selected.

2.3.2.2.2 PF2 Key Context-sensitive Help

Provides context-sensitive help about a specific item, field, or window.

2.3.2.2.3 PF3 Key Exit

2.3.2.2.4 Exit is defined in 2.3.2.1.1 above.PF4 Key Backtab

Moves the cursor to the previous entry field. The cursor moves from right to left, bottom to top.

2.3.2.2.5 F10 Key Menu Bar

Moves the cursor to the menu bar, if one is available, at the top of the window or pop-up currently in focus.

2.3.2.2.6 F12 Key Cancel

Cancel is defined in 2.3.2.1.3 above.

2.3.2.2.7 TAB Key Tab

Moves the cursor to the next entry field. The cursor moves from left-to-right, top-to bottom.

2.3.2.2.8 PF1,H Key Sequence

Application Help. Provides information about the particular segment of the application being used.

2.3.2.3 Lock

If a user is waiting for a lock that times out, then appropriate notification should be given to the user.

2.3.3 User Interface Conventions for GUI Mode

2.3.3.1 GUI Standards Document

VistA/RPMS packages should follow the guidelines for GUI applications set forward in the VistA/RPMS Standards document. See Appendix G.

2.4 OPERATING SYSTEM PROGRAMS, SCRIPTS AND FILES STANDARDS

2.4.1 Purpose

To provide guidance in certifying and distribution Operating System Programs, Scripts and Files that support the RPMS MUMPS applications. Operating Systems will include UNIX, Windows NT and Windows 2000.

2.4.2 Standards

2.4.2.1 Namespacing

All Operating System Programs, Scripts and Files shall be namespaced with the assigned namespace (i.e., aibxxxx).

2.4.2.2 Directory

All applications-related Shell Programs and Files shall reside in a unique subdirectory under a variable, parameter-driven directory depending on the operating system and user input. The subdirectory will be named using the assigned namespace (e.g., /usr/aib or D:\usr\aib).

2.4.2.3 Version

The first line of all Operating System Programs, Scripts and Files shall contain the name of the Shell Program or File and the version number of the application. The version number shall be the same as the associated MUMPS application version number.

2.4.2.4 Install Shell Program

An install Program shall be included in all distributions. This Install Program shall perform the following:

2.4.2.4.1 OS Determination

Determine the Operating System type to insure that correct files are loaded during the installation

2.4.2.4.2 Obtain Variables

Obtain variables/parameters based on user input to drive the actual installation

2.4.2.4.3 Ownership

Ensure that the ownership of the Operating System Programs, Scripts and Files are properly set and that they are correct for the Operating System.

2.4.2.4.4 Group Membership

Ensure that the group membership of the Operating System Programs, Scripts and Files are properly set and that they are correct for the Operating System.

2.4.2.4.5 Modes

Ensure that the modes of the Operating System Programs, Scripts and Files are properly set.

2.4.2.4.6 Subdirectories

Ensure that all Operating System Programs, Scripts and Files reside in the proper subdirectories based on variable, parameter driven input.

2.4.2.5 Operating System Commands/Utility Files

The RPMS application packages submitted for certification should not include standard OS commands and utility files (e.g., sendto command).

2.4.2.6 Patches

All corrections of errors identified in the production release should be treated similar to corrections in the associated MUMPS application by using the IHS Patch System. Each patch should be documented in the Operating System Shell Program, script or File beginning with the second line.

2.4.2.7 Enhancements/Modifications

All future enhancements and modifications of the application should be coordinated with the person who is responsible for developing or maintaining those applications affected.

3 APPENDIX A - RPMS NAMESPACE ASSIGNMENTS

3.1 PURPOSE

Systems developed for the RPMS are identified by a set of program and file names, and are assigned a range of file numbers with the computer. Standards must be followed in assigning these names and numbers to ensure that they are unique and do not overlap between systems. The purpose of this documentation is to define these standards.

3.1.1 Responsibility

3.1.1.1 Program/File Name Assignments

The DBA will maintain a master list of all RPMS name assignments, and will assign and/or approve of new names for new systems.

3.1.1.2 File Numbers

Blocks of file numbers have been allocated to RPMS core systems, the Division of Data Processing Services (DDPS), and to each Area, as indicated in Figure 2.

The DBA will be responsible for maintaining and assigning file numbers for RPMS applications; the Area ISCs will maintain and assign file numbers for local systems.

3.2 GENERAL STANDARDS

The general standards to be followed in assigning names are as follows:

3.2.1 Naming Assignments

3.2.1.1 Namespace Position 1

The namespace of all IHS-developed packages assigned after May 5, 1993 must begin with B.

3.2.1.2 Position 2-3

The next two to three characters will be the system prefix, and will be used to identify the application system itself (i.e., CHS for Contract Health Services, or CA for Cost Accounting).

3.2.1.3 Remaining Positions

The remaining characters will be used to identify names used within the application.

3.2.2 Other Standards

3.2.2.1 All Names

The use of "Z" as the fourth or fifth character, immediately after the system prefix, to designate locks, to distinguish locks from menus and options.

3.2.2.2 XB/ZIB - Utilities

XB/ZIB will be assigned to utilities that have applicability for multiple systems.

Initially, there will be no central assignment of XB/ZIB names. A developer will only need to select a name not already being used. If this should become a problem later, the DBA will coordinate assignments of XB/ZIB names. Routines that only apply to a particular application should carry the name prefix of that application.

3.2.2.3 BZ - Local Area Routines

BZ will be reserved for local program development, when there is no immediate plan or intention to distribute the system outside of the Area. ISC's should notify everyone within their Area who may be writing programs to use BZ as the first two characters of their application name. The ISC's should monitor and/or assign names with the BZ format to assure there is no duplication within the Area.

It is recommended that Areas use the third digit to identify the Area in which the system was developed. In this way, you can ultimately share the program with another Area with the assurance that it will not interfere with the local programs developed by the other Area. A list of Area codes to be used for this purpose is shown in Figure 1. BZ should be used when there is no intention to share the system outside of the Area. In all other cases, the ISC should contact the DBA for a global name assignment.

3.2.2.4 Universal Globals

The DBA will coordinate the name assignments of globals that have applicability for multiple systems. Globals that only apply to a particular application should carry the namespace of that application.

3.2.2.5 Standard Table Globals

The first two characters of the IHS Standard Tables will begin with AU. The third character will indicate the type of global as follows:

T = Table

P = Patient File

A = Administrative File

The fourth character will indicate whether the global supports UCI translation (i.e., can be accessible from multiple UCI's). These codes are:

T = Translation required

N = Not required

Thus a table, supporting UCI translation, would have as its first four characters:

AUTT _ _ _ _

Care should be taken not to misuse the AU name. Routines and globals that only apply to a particular system should carry the name prefix of that system.

3.2.2.6 Use of Prefix AVA

This prefix is reserved for IHS changes to VA files such as the New Person file. Its use should be coordinated through the DBA.

Codes for Area Use in Local Program Development	
A	Alaska
B	Billings
C	Aberdeen
D	Bemidji
H	ITSC
L	California
N	Navajo
O	Oklahoma
P	Portland
Q	Albuquerque
S	Tucson
U	Nashville
X	Phoenix

Figure 1

RESERVED FILE NUMBERS

File Number Range	Reserved For
0-9999	VA VistA Supported Systems
90000-99999	IHS RPMS Systems (After 5/5/93)
8000000-8999999	SET, DSD, DTM, and DDPS
9000000-9999999	IHS RPMS Systems (Pre-1993)
1000000-1099999	Area 10, Site 00-99, File 000-999- Aberdeen
1100000-1199999	Area 11, Site 00-99, File 000-999- Alaska
1200000-1299999	Area 12, Site 00-99, File 000-999- Albuquerque
1300000-1399999	Area 13, Site 00-99, File 000-999- Bemidji
1400000-1499999	Area 14, Site 00-99, File 000-999- Billings
1500000-1599999	Area 15, Site 00-99, File 000-999- California
1600000-1699999	Area 16, Site 00-99, File 000-999- Nashville
1700000-1799999	Area 17, Site 00-99, File 000-999- Navajo
1800000-1899999	Area 18, Site 00-99, File 000-999- Oklahoma
1900000-1999999	Area 19, Site 00-99, File 000-999- Phoenix
2000000-2099999	Area 20, Site 00-99, File 000-999- Portland
2100000-2199999	Area 21, Site 00-99, File 000-999- Tucson

Figure 2

3.3 FILE NUMBERING CONVENTIONS

3.3.1 Reserved File Numbers

Blocks of file numbers have been reserved for each Area, the DDPS, and for RPMS core systems as shown in figure 2 above.

3.3.2 Multiple Files

When applications are developed that involve multiple files, the same integer may be used for all files directly related to the package, and decimal numbers used for members of the group. For example, a Personnel Package developed in Area 10, Site 05, might have the file number 1005007 and the group of files might be numbered as follows:

Personnel	1005007.1
Title	1005007.2
Wage Scale	1005007.3

3.3.3 Common Pointer Files

Common pointer files that are pointed to by various applications (e.g., LOCATION file) should be numbered in a different manner to indicate these files are not unique to a single application. For example, the common pointer files in the IHS RPMS Systems will be numbered 9999999.n where n is the next available sequential canonic number. If you have more than nine files in any single group you should append two digit numbers if you want them to sort out properly (e.g., 01, 02)--(.11 sorts lower than .2).

3.3.4 Area ISCs Files

The Area ISCs will assign file number ranges to their various sites, keeping in mind that there may be more than one system at a site. A range of numbers should also be retained for the Area office. The allocation of numbers will vary from Area to Area, depending on circumstances. Not that one may typically want to assign different site numbers to FileMan globals in PRD and PVT UCI's.

3.3.5 Locally Developed Application

When a locally developed application is to be incorporated into the IHS Core System, the files will be renumbered with the range 9000000 - 9999999. If one site wants to incorporate another site's application, the receiving site may want to renumber the files, or it may choose to run the application with the original site's file numbers. When each site conforms to these conventions, the transfer of applications from site to site should not be difficult.

3.3.6 File Manager File Numbers

File numbers in File Manager must be canonic; i.e., must not have leading zeros or trailing zeros following a decimal. This includes sub-files generated by multi-valued fields.

4 APPENDIX B - REQUEST FOR EXEMPTION TO RPMS PROGRAMMING STANDARDS

Request for Exemption to RPMS Programming Standards

Package: Date:

Program:

Line Number:

Applicable Standard:

Reason for Exemption:

Developer

SRCT Review Date:

Recommend APPROVAL___ DISAPPROVAL___

Comments:

Verifier(s)

SET Action Date:

Request APPROVED___ DISAPPROVED___

Comments:

Director, DSD

5 APPENDIX C - UNIX/DOS FILE NAMING STANDARDS

5.1 PURPOSE

Applications developed for the RPMS are distributed in sets of files, the names of which must conform to the specifications of the host operating system. A defined operating system-independent file name format will simplify the activities of staff responsible for manipulation of these files (Area support staff, facility site managers, etc.) and provide a structured paradigm that can be used for automated manipulation of these files by future applications. This standard is being established for this purpose.

5.1.1 Distributed Applications

This standard applies only to files of distributed applications. It is not necessarily binding upon filenames for test and verification copies of an application, due to their inordinately long version numbers (See Appendix D - Version Numbering for RPMS Systems).

5.2 GENERAL STANDARDS

The general standards to be followed in assigning names for all host operating system files are as follows. Patches are addressed separately.

5.2.1 Positions 1-4

Positions 1-4 are reserved for the namespace of the RPMS application. If the application namespace is less than four characters, the remainder of the character positions may be padded with underline (_) character(s).

5.2.2 Position 9

Position 9 is a period or decimal point (.).

5.2.3 Namespace Case

Namespace and alphabetic codes will be in lower case for Distribution filenames.

5.3 STANDARDS FOR DISTRIBUTION FILES

The standards to be followed in assigning names for RPMS application distribution files are as follows.

5.3.1 Positions 5-6

Positions 5-6 are reserved for the major portion of the RPMS application's version number (the portion preceding the decimal point). If less than two digits, this value should be right-justified and padded with one or more number 0 digits.

5.3.2 Positions 7-8

Positions 7-8 are reserved for the minor portion of the RPMS application version number (the portion following the decimal point). Position 8 will always be a 0 except when annotating a .pdf file as outlined elsewhere.

5.3.3 Position 10

Position 10 is reserved for an alphabetic code indicating the type of distribution file. The codes to be used are as follows:

Code	Description
g	Distribution Globals
u	Unix Files. Archived Unix Files
k	A file containing a KIDS transport distribution
z	Zipped/ Windows-related file

Example: apch0190.k would denote production UCI KIDS distribution for Version 1.9 of the PCC Health Summary package.

5.3.4 Position 11

Position 11 is a flexible field. Possible values for this field are as follows.

Code	Description
m	Routines or globals to be installed in MGR UCI
s	Required if using "u" in position 10 to denote Unix scripts/files

Example: xu_0710.rm would denote VA Kernel Version 7.1 routines to be installed in the manager UCI

5.3.4.1 Multiple Files

A single-digit numeric value used to denote one of multiple files to be installed in a production UCI.

Example: ade_0520.gl would denote the first set of globals to be considered in an installation of Version 5.20 of the IHS Dental package

5.3.5 Position 12

Position 12 is an optional numeric field. Possible values are as follows:

5.3.5.1 Multiple MGR/Script Files

A single digit value used to denote one of multiple files to be installed in the manager UCI or multiple UNIX script files.

Example: xu_0710.gm2 would denote the second set of manager UCI globals to be considered in an installation of Version 7.1 of the VA Kernel

5.4 STANDARDS FOR PATCH FILES

The standards to be followed in assigning names for RPMS application patch files are as follows.

5.4.1 Position 10

Position 10 is reserved for a numeric. The numeric is to be the "tens" position of a two digit patch number. If there is no ten digit, a zero is to pad this space.

Example: xu__0710.10k would denote routines file for patch number 10 of Kernel Version 7.1

Example: xu__0710.02k would denote routines file for patch number 2 of Kernel Version 7

5.4.2 Position 11

Position 11 is reserved for a numeric. The numeric is the "ones" position of a two digit patch number.

Example: xu__0710.01k would denote a routines file for patch number 1 of Kernel Version 7.1

5.4.3 Position 12

Position 12 is reserved for an alpha-character as follows.

Code	Description
b	Patch Globals
n	Patch Notes
k	File containing a KIDS transport

Example: bw__0200.01n Notes for patch 2 of Women's Health Version 2.0.

5.5 STANDARDS FOR HOST OPERATING SHELL PROGRAMS

Shell programs shall be in the form as outlined above through position nine. Position 10-16 will contain "install".

Example: aib_0300.install

5.6 STANDARDS FOR DOCUMENTATION FILES

The standards to be following in naming documentation files is as follows.

5.6.1 Position 8

Character position 8 in a documentation file will contain one of the following codes.

Code	Description
u	User Manual
t	Technical Manual
s	Security Manual
r	Readme File
i	Installation Guide
n	Release Notes
o	Other

5.6.2 Position 10-12

Character positions 10-12 will contain "pdf" to designate that the particular manual was prepared in "pdf" format.

- Example: bw__020u.pdf - User Manual in PDF format for the Women's Health Version 2.0
- Example: bw__020t.pdf - Technical Manual in PDF format for the Women's Health Version 2.0

5.6.2.1 Distributed Documentation File

All documentation files will be "zipped" prior to final distribution. The "zipped" file will contain those files as outlined above.

- Example: bw__0200.zip will contain the User, Technical and Readme File for Women's Health Version 2.0

5.7 STANDARD FOR FINAL DISTRIBUTION FILE

The final distribution file will be a compressed tar file named using the above general standards for Positions 1-4 and position 9. The file will contain all the files necessary

for the application, i.e., routines, globals, notes, Readme, all documentation files and any other files specified.

- Example: bw__0200.tar.gz contains:

bw__0200.k	KIDS transport file
bw__0200.g	Globals
bw__0200.zip	Archived files (Windows)
bw__0200.us	Archived Host Operating System Files
bw__0200.install	Host Operating System Installation Program Files

6 APPENDIX D - VERSION NUMBERING FOR RPMS SYSTEMS

6.1 PURPOSE

Applications developed for the RPMS go through three primary phases of evolution; testing (both alpha and beta), verification, and distribution. A single version of an application can go through multiple iterations of the first two steps before being distributed, with each iteration differing from the previous one. This standard establishes a standard format whereby multiple iterations of an application can be easily managed by developers, testers, and verifiers.

6.2 DEFINITIONS

6.2.1 Distribution Version

The version number assigned to the application when released for IHS-wide installation.

6.2.2 Target Version

The intended distribution version number assigned to the application while going through the testing and verification phases.

6.2.3 Iteration

A single set of files containing all routines, globals and notes necessary to install the application.

6.2.4 Sequence Number

Tthe number identifying the current iteration of the application in either testing or verification phases. Testing sequence numbers and verification sequence numbers are not consecutive.

6.3 FORMAT

6.3.1 Format of UNIX file name:

6.3.1.1 Testing

Versions of an application submitted for alpha or beta testing will be signified by an lowercase letter t immediately following the target version. Immediately following the lowercase letter t will be the sequence number of the test version. There is no distinction between alpha and beta test iterations.

- Example: bw__0350.t3r would identify the third test iteration of version 3.5 routines file of Women's Health.

6.3.1.2 Distribution

Applications that have passed verification will assume the target version of the last verification iteration.

- Example: If bw__0350.t3r passes verification, the application will be distributed as version 3.5 and will be annotated in the unix file as bw__0350.r.

6.3.2 Format of M Routines

6.3.2.1 Second Routine Line

The 2nd line of all package M routines will reflect the test iterations.

- Example: ABCTest; IHS/ABDEV/MMM-Example M routine;
  11/4/95;;1.0t3;Test;*0*;11/4/95

6.3.2.2 Package File

The Package File will reflect the test iterations in the current version field. When the package passes verification the Package File will be cleaned to only include the final distributed version number.

7 APPENDIX E - STANDARDS FOR SUBMISSION OF DATA TO NPIRS

7.1 OVERVIEW

Many applications being developed to run on facility computers will have a requirement to generate and transmit data into an IHS-wide reporting system maintained at the NPIRS, in Albuquerque, New Mexico. Examples are the PCC Data Entry System, Patient Registration, Dental Data System and the Admission, Discharge and Transfer (ADT) System.

For these systems, the data will flow from the facility to the Area, where data will be consolidated for all facilities for each system, and then forwarded periodically via IHS wide-area network for each system to the NPIRS. The NPIRS will not accept data directly from individual facilities.

Standards and procedures have been developed to facilitate this transfer, and are described in this document.

7.2 FACILITY PROCEDURE

7.2.1 Creation of Transaction Global

A programmer or analyst developing a system with IHS-wide reporting requirements will need to determine the criteria by which data will be selected from the facility data base for transmission to the Area/NPIRS.

7.2.1.1 Process Options

There are a number of ways this can be done, including the following:

- Selection based on the transaction encounter date,
- Selection based on the facility posting date,
- A special flag to indicate whether data has been transmitted,
- Creation of a special global identifying records that have been created or modified since the last transmittal.

However, the data is identified, a program will be required to extract the data from the data base concerned, and generate a global that can be written to the host operating system (or transmitted directly) to the Area.

7.2.1.2 Global Creation Standards

The standards to be used in creating this global are as follows:

7.2.1.2.1 Global Name

The name of the global will be the four-digit namespace assigned to the system, such as AAPC, ACHS, BCHR, etc., concatenated with the word "DATA". If the namespace has less than 4 digits assigned, characters should be added to fill out the four spaces. Examples of these names are:

^AGTXDATA - Patient Registration

^AAPCDATA - PCC Data Entry

^ADENDATA - Dental Data System

7.2.1.2.2 "0" Node

The global will have a "0" node, followed by a series of data records subscripted from "1" to "n" for that transmission. For example:

- ^AGTXDATA(0)=
- ^AGTXDATA(1)=
- ^AGTXDATA(2)=

7.2.1.2.3 Pieces 1-5 and 7

The format of the "0" node is illustrated in Figure 1. Pieces 1-5 and 7 are required. The pieces are as follows:

7.2.1.2.3.1 Facility code

This is the standard IHS 6 digit code identifying the facility.

7.2.1.2.3.2 Facility Name

Narrative name of the facility.

7.2.1.2.3.3 Date of Run

The date that the global was created in the format YYYMMDD, where YYY is the number of years since 1700; i.e. 1988 would be 288, MM is the month (01-12), DD is the day of the month.

7.2.1.2.3.4 Beginning Date.

This field is required, but the definition of the field is program specific. For many programs, data will be selected by a range of dates (i.e., posting dates), and this field will be the earliest date in the date range. If this date is not important or used by the program concerned for data extraction, the date can be the date of the run.

7.2.1.2.3.5 Ending date

See above. If not important to the data extraction, the date can be the date of the run.

7.2.1.2.3.6 Last Record Transmitted

This field is optional, and was intended for use in cases where data was extracted based on some type of sequential number.

7.2.1.2.3.7 Number of Records

This is a count of the total number of records contained in the global (not including the "0" node), and is required.

7.2.1.2.3.8 Cartridge Number

The number of the cartridge on which the global will be written. This is for facility audit purposes. Not required.

7.2.1.2.3.9 Date Transmitted

The date the data file was transmitted. This field is normally not used, since this date and the date of the run are usually the same.

7.2.1.2.4 Data Node Format

The format of data in each data node is as follows:

Globalname(n)=xxx^data string (fields separated by a "^")

where: n is the next sequential subscript for the transmission,

xxx is the transaction type, i.e., RG1, RG2, IR1, etc.,

data string is the actual data being transmitted. Each field is separated by the "^". A piece must exist for each field in the transaction, regardless of whether there is any data for that piece, and all pieces must be in the same sequence as the fixed length transaction required at NPIRS.

Care must be taken to assure that the total length of the data in the node will never exceed 511 characters in length. If this should ever happen, the data needs to be broken down into two record types (i.e., RG1 and RG2).

An example of a global created for a particular application might be as follows:

```
^AGTXDATA(0)=508201^CARL ALBERT
HOSPITAL^2860804^2860701^2860731^^^3^^^
^ATGXDATA(1)=RG1^508201^336^SMITH^JAMIE^H^01^0608908^..
^AGTXDATA(2)=RG2^TOAHTY^MARY^^^Y^223098761A^...
^AGTXDATA(3)=RG1^508201^947^BEGAY^JOHNATHON^L^01^02
14893^...
^AGTXDATA(4)=RG2^ADAMS^EVA^^^N^123456789A^...
```

The AIB Record Consolidation System at the Area will eventually receive the above data and process it for the central computer. If there are two record types, as in the above example, they will be combined into a single record at NPIRS, with the data from the second record (RG2) added onto the end of the data from the first (RG1).

7.2.2 Transmission of Global to Area

A general purpose utility routine XBGSAVE has been developed to read a global as described above, and copy it to a tape cartridge. This utility requires that the name of the global be placed in the variable XBGZL prior to execution, i.e.,

S XBGL="AGTXDATA:

D ^XBGSAVE

Other variables can be set to select various IO options. See XBGSAVE for details.

Execution of this routine can be initiated automatically by incorporating the above routine at the end of the program that creates the global, or can be designed as a

separate option to be selected from the program's main menu.

For MSM Systems:

The XBGSAVE routine will create global save format to a UNIX text file.

When the copy operation has been completed, the original global needs to be deleted before additional data is extracted and posted from the facility database. If an operator fails to do this, new data will be merged with the data already sent to the Area, and this will all be re-forwarded to NPIRS on the next transmission. This could cause problems, depending on the system concerned. A programmer might want to consider automatically killing the global after successfully copying to the host operating system.

The above process can be done at whatever frequency is agreed upon between the Area and the facility.

7.3 AREA PROCEDURE

When facility data files are received at the Area for a particular application, they are merged into a file containing similar data from other facilities with the utility routines "AIB".

The AIB consolidate routines will determine the name of the file to be updated from the name on the incoming file (i.e., AGTXDATA). If the file already exists on disk (i.e., AGTXBLOB), the AIB consolidation routines will merge the data from the incoming data file into the global file. If the file does not exist, it will create the file.

At periodic intervals, normally once or twice a month, the Area will create a transmission file for all information in the global files using the AIB routines. This will delete the global files on the Area disk, and the cycle will start over when the next data file is received from a facility.

These convert programs refer to a table that identifies the fixed length transaction to be created from the incoming data records. The pieces in the data string must be in the same sequence as the transaction to be created. The table identifies the column in which each field starts, and the length of the field. For instance:

```
;1;3;   RECORD TYPE      (Starts in col.1;3 characters long)
;4;6;   IHS FACILITY CODE   (Starts in col.4;6 characters long)
;10;2;  TYPE OF ACTIVITY
;12;5;  REFERRAL CODE
```

If an incoming field is longer than the field in the fixed length transaction, it will be

truncated.

If more than one record type is contained in the incoming global, e.g., RG1 and RG2, the convert program will combine the two records into a single transaction by adding the data from record two (RG2) onto the end of record one (RG1).

The fixed length transactions will be written out to a transaction tape at NPIRS, or stored in a transaction file, and processed on the next update of the application concerned.

Any programmer/analyst designing a system with reporting requirements to NPIRS will need to notify the SET with as much lead time as possible to allow the convert program to be developed by SET staff.

Each Area will be required to transmit monthly data sets to NPIRS computer. Users can submit their data using the Area Data Consolidation System (AIB).

7.4 NPIRS PROCEDURE

The process for updating the master database will revolve around a program called *incoming* that begins by identifying file types and processing order of files received, updating each subsystem in the master database, activating a report generator, and electronically informing the submitting area of the status of its data.

7.4.1 IHS Subsystems

The master database is comprised of several systems that include Patient Registration (GTX), Patient Eligibility (ELG), Health Record Add (GHA), Inpatient Care (APC), Outpatient Care (INP), Inpatient and Outpatient Contract Health Services (CHS), and Patient Merge (GDM). The systems are listed in the order in which they are processed.

7.4.1.1 Patient Registration (GTX)

This subsystem is the first to be processed since it adds patients' records that are updated by the rest of the systems listed. The master database is searched for an existing facility code and health record number. If an entry is found, the health record is scanned for the associated patient record link. If no link is found, this indicates that this record was inserted by the health record add (GHA) subsystem and needs to be updated with the rest of the patient's personal information. If an entry is not found, the patient and associated personal information is inserted.

7.4.1.2 Patient Eligibility (ELG)

This subsystem is the second to be processed due to the remaining subsystems requiring updated eligibility information. This system updates an existing patient's eligibility record or creates a new eligibility record, depending on whether a starting or ending data is included in the incoming record.

7.4.1.3 Health Record Add (ELG)

This subsystem is the third to be processed because the remaining subsystems update records according to facility code and health record number. This system inserts a new record into the chart table with only a health record number and an associated facility code. A flag is set in the new chart record to signal the GTX subsystem to connect this health record with a person. If the associated patient record exists, the GTX system updates the patient record with GTX information and the flag is removed.

7.4.1.4 Ambulatory Patient Care (APC)

This subsystem, which inserts an outpatient record, is the fourth to be processed.

7.4.1.5 Inpatient Care (INP)

This subsystem, which inserts an inpatient record, is the fifth to be processed.

7.4.1.6 Contract Health Services (CHS)

This subsystem, which inserts a contract health services inpatient or outpatient record, is the sixth to be processed. The source of this information not only comes from the Areas but from Blue Cross/Blue Shield (BCBS).

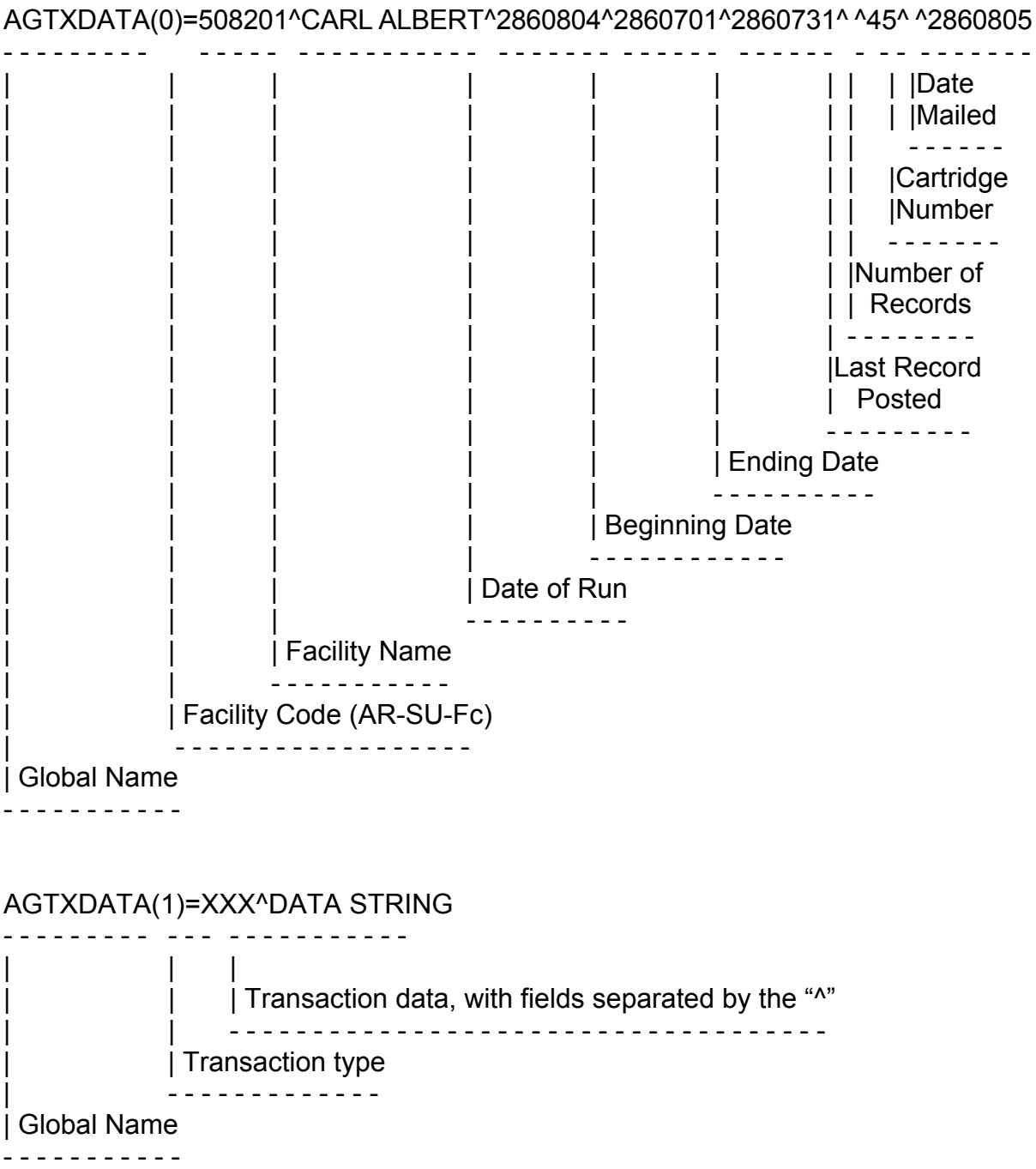
7.4.1.7 Patient Merge (GDM)

Formerly Delete/Merge, this subsystem is the last to be processed. This subsystem closes out old health record numbers by setting an ending date in the record, and creating a new health record using the updated information provided and populating the rest of the new record with the most recent information available.

7.4.2 Post Updating

To close out a month's processing, a table containing each Area's submission information is updated with the status of each system's results. If an Area has not submitted its data, the table will contain NULL values that will flag the operator and send a message to the Area contact and the appropriate NPIRS individual(s). A log file is also created for every file processed and resides in the same directory as the processed file that contains a full account of the processing and a summary of the essential totals. These totals are inserted into the Area's submit table that informs NPIRS that an Area has submitted its data. This process also allows for a previous month's count to be compared to the current month's count. If the current counts are within a preset percentage, an error is generated and sent to the Area contact and NPIRS individuals.

FORMATS OF GLOBALS CREATED AT FACILITY



8 APPENDIX F - RPMS DOCUMENTATION STANDARDS

8.1 PURPOSE

The purpose of these documentation standards is to provide a basic documentation structure that can be applied to every software package, to provide consistency in all documentation, and to provide criteria by which documentation of a national package can be verified.

8.2 GENERAL STANDARDS

The general standards to be follow in preparation of all RPMS documentation is as follows.

8.2.1 Definitions

8.2.1.1 RPMS Package

An RPMS package is RPMS software intended for IHS and tribal distribution and implementation that is fully supported by the DSD. Each package is considered a component of RPMS and is assigned by the Director, DSD, to a development center for development and maintenance.

8.2.1.2 Non-RPMS Automated Information Systems Package

A software package not developed by an approved development center, and not supported by DSD, but may be for use within IHS.

8.2.1.3 Package Documentation

Package documentation is the information that describes the functions, implementation, use, maintenance, and distribution of RPMS packages.

8.2.1.4 Sensitive Information

Sensitive information is information that requires protection due to the risk and magnitude of loss or harm that could result from inadvertent or deliberate disclosure, alteration, or destruction.

8.2.1.5 Adobe Acrobat Reader/Exchange

Commercial software designed to bring electronic documents to a wide range of users. Cross-platform documents, which are created in Adobe Acrobat Portable Document Format (PDF), are called PDF documents. Acrobat Reader enables Windows and Windows NT, Macintosh, DOS, and UNIX users to review, navigate through and print any PDF document.

8.2.2 Mandatory Components

The four mandatory components of RPMS software documentation consists of:

- Installation Guide and/or Release Notes
- Technical Manual
- User Manual
- Security Manual, if applicable

8.2.3 Preparation

All RPMS software documentation will be prepared in electronic format using the Adobe Acrobat software (pdf format). The pdf file will be distributed as part of the national release and the file will reside as part of the archived distribution file.

8.2.4 Documentation Descriptions and Standards

8.2.4.1 Installation Guide (Mandatory)

8.2.4.1.1 Title Page

Identify the package by name, version number, and release date.

8.2.4.1.2 Installation Issues

Provide an instructional guide for installing the software. Describe issues that should be considered prior to initialization and how to prepare for initialization.

8.2.4.1.3 Installation Instructions

Describe the installation process in logical steps and include a statement recommending where the software should be loaded (e.g., "initially load software into a test account and then finally into the production account"). Note any routines, globals, etc., that may be removed from the system after completion of the installation.

8.2.4.1.4 Installation Configuration

Provide guidance and suggestions for system configuration and global placement. List any requirements necessary for successful installation of the package. List any reference material that may be required during the installation process. List items that the installer should produce from the system after installation of the national package.

8.2.4.1.5 Required Resources

Describe the resources required for the national package. Include Central Processing Unit (CPU) capacity, disk space, unique devices, and other pertinent resources. Provide a formula for sizing, if applicable.

8.2.4.2 Release Notes

Mandatory for subsequent releases - may be included as part of the Installation Guide but must be distinguished as such.

8.2.4.2.1 Title Page

Identify the RPMS package by name, version number, and release date.

8.2.4.2.2 Modifications and Enhancements

Describe any modifications and enhancements to the national package software since prior release. This information is needed in advance of loading the software.

8.2.4.3 Technical Manual

8.2.4.3.1 Purpose

Technical documentation should provide sufficient information about the software for programmers, ISCs and ITSC technical personnel to operate and maintain the program applications without additional assistance from the package developer(s). The Technical Manual must contain, at a minimum, the mandatory sections listed below.

8.2.4.3.2 Title Page (Mandatory)

Include the name of the national software package, the version number, the preparation date, the name of the development center and the logo "IHS RPMS".

8.2.4.3.3 Preface (Mandatory)

Supply a brief statement identifying the document in terms of its purpose, scope and targeted audience.

8.2.4.3.4 Table of Contents (Mandatory)

Provide a table of contents with page references to major chapters and/or sections of the manual.

8.2.4.3.5 Introduction (Mandatory)

Include an overview that describes the package. The introduction should convey to the reader the major function(s) and purpose(s) of the package, and how the software accomplishes the objective(s).

8.2.4.3.6 Orientation (Optional)

Address package-specific notations or directions (e.g., symbols used to indicate terminal dialogues or user responses).

8.2.4.3.7 Implementation and Maintenance (Mandatory)

Provide information to assist the ISCs, ITSC personnel, and PSGs in the implementation and maintenance of the national package. This section may include

information regarding the entry of required site-specific data; a description of parameters configured to meet the needs of individual sites; sample configurations; and work sheets to assist in determining the parameters to be entered for the site.

8.2.4.3.8 Routine Descriptions (Mandatory)

Provide a list of routines with comprehensive descriptions of the function.

8.2.4.3.9 File List (Mandatory)

Include a list and brief description of files that come with the package. The description should indicate what data comes with the file and whether or not that data will overwrite existing data, if applicable (this will normally apply only to VA packages since including data with a file is against IHS Standards).

8.2.4.3.10 Exported Options (Mandatory)

Provide a list of the options in the package. Indicate distribution of menus to users and note any restrictions on menu distribution.

8.2.4.3.11 Cross-references (Mandatory)

Provide a brief description of all cross-references exported with the package.

8.2.4.3.12 File Diagram/Flowchart (Optional)

For packages that include numerous files, provision of a chart representing the relationship among files is highly desirable.

8.2.4.3.13 Archiving and Purging (Mandatory)

Describe any archiving or purging capabilities of the package and any necessary instructions or guidelines.

8.2.4.3.14 Callable Routines (Mandatory)

List all entry points in the package that can be called by other applications. This list must include the actual entry points, a brief description of the function of these entries, a description of all required variables, and any restrictions on the use of the entry points.

8.2.4.3.15 External Relations (Mandatory)

Explain any special relations and agreements between the routines and/or files/fields in this package and the routines and/or files/fields in other packages. List any routines essential to the functions of this package, for example: Could an outpatient facility function without programming related to inpatient activity and avoid system failure? Specify the version of VA FileMan, VA Kernel, and other packages required to run the package.

8.2.4.3.16 Internal Relations (Mandatory)

Identify, if applicable, any routines, files or options within this package that cannot function independently of other programs. For example, Which menus can stand alone? Does the functioning of a particular option assume that entry/exit logic of another option has already occurred? List such options with their programming SACC approval dates.

8.2.4.3.17 How to Generate On-line Documentation (Mandatory)

Provide the file numbers and/or file number ranges, and namespaces along with any special templates. Inform the users where to find the Kernel documentation and how to print the data dictionaries and menu diagrams.

8.2.4.3.18 Glossary (Mandatory)

Provide a glossary of terms that relate to the specific national package.

8.2.4.3.19 Index (Optional)

Provide a package-specific index.

8.2.4.3.20 SAC Requirements/Exemptions

Provide a listing of any items required by the SAC to appear in the Technical Manual. All exemptions granted by the SACC must noted specifying the date of the exemption and the actual exemption.

8.2.4.4 User Manual

8.2.4.4.1 Purpose

The User Manual is a document designed to be helpful to the user. This documentation will provide sufficient information for users to competently operate the national software package. Variability in the content is accepted. The User Manual must contain, at a minimum, the mandatory sections listed below.

8.2.4.4.2 Title Page (Mandatory)

Include the name of the software package, the version number, the preparation date, the name of the Development Center, and the logo "IHS RPMS".

8.2.4.4.3 Preface (Mandatory)

Supply a brief statement identifying the document in terms of its purpose, scope, and targeted audience.

8.2.4.4.4 Table of Contents (Mandatory)

Provide a table of contents with page references to chapters and/or sections of the manual.

8.2.4.4.5 Introduction (Mandatory)

Provide an overview that sets forth a description of the software package. Note related RPMS and/or VA manuals and other reference materials for a modular manual and the purpose for the individual module. Distinguish the major topics and issues within the package. The introduction should convey to the reader the major function(s) and purpose(s) of the package, and how the software accomplishes the objective(s).

8.2.4.4.6 Orientation (Optional)

Address any package-specific notations or directions (e.g., symbols used to indicate terminal dialogues or user responses).

8.2.4.4.7 Package Management (Mandatory)

Address unique legal requirements pertaining to the package and necessary security measures to protect the integrity of the package and its data (e.g., a package may use an electronic signature code or data that may not be changed because it is supplied by another agency).

8.2.4.4.8 Package Operation (Mandatory)

Describe what the user needs to know in order to competently operate the package. The information should include how the user can access on-line documentation.

8.2.4.4.9 Glossary (Mandatory)

Provide a glossary of terms that relate to the specific package.

8.2.4.4.10 Index (Optional)

Provide a package-specific index.

8.2.4.5 Package Security Manual or Guide

8.2.4.5.1 Purpose

A package security manual or guide will be created for controlling the release of sensitive information related to the national software package. This document will not be included in any Freedom of Information Act (FOIA) request releases. Distribution of this document is limited to the ISC and ITSC personnel. Since certain keys and authorizations must be delegated for proper management of the system, information about these items may be found elsewhere in the technical and user manuals.

8.2.4.5.2 Title Page (Mandatory)

Include the title "Package Security Guide", the name of the software package, the version number, the name of the Development Center, the preparation date, the logo "IHS RPMS" and the words "SENSITIVE INFORMATION".

8.2.4.5.3 Package Security (Mandatory)

Provide a section on security requirements for the national package. Document all locks and keys, special FileMan access codes, and other security measures included in the package. Describe the purpose of security keys. Include official policy unique to the package regarding the modifications of software and distribution of the package.

8.2.4.6 Users Guide to Computing

The *Users Guide to Computing* has been adopted as a stand-alone instructional guide for general computer usage. This guide describes programming conventions common to all national packages and is to be used as a reference source.

8.2.4.7 Patch Documentation Requirements

8.2.4.7.1 Notes

All patches to certified RPMS software require a notes file outlining system requirements, contents of the distribution, modifications to the software, reason for the patch, etc. Developers should follow the format outlined in Notes File procedure elsewhere in this handbook.

8.2.4.7.2 Patch Module Entry

Each patch to a certified RPMS package will be entered into the IHS Patch Module by the responsible developer. In addition, another developer is responsible for completion of the patch prior to review and verification by SRCT.

8.3 DOCUMENTATION STYLE STANDARDS

The following style standards are provided to promote consistency in IHS RPMS package documentation.

8.3.1 Abbreviations and Acronyms

Spell out the abbreviation or acronym when it is first used and put the abbreviation or acronym in parentheses after it is initially spelled out (e.g., Indian Health Service (IHS)). After initially spelled out, use the abbreviation or acronym without parentheses.

8.3.2 Appendices

Appendices are considered supplemental information and should be used for appropriate materials such as listings of descriptions or definitions of data elements, sample reports generated by the package, summaries of commands and biographies of supportive reference materials.

8.3.3 Change Pages

Change Pages will be identified with the new version number and release date.

Additional distinguishing marks (e.g., vertical lines in margins) can be used as needed. Distribution of change pages should be coordinated with the SRCT.

8.3.4 Computer Dialogue

Recreate the program's computer dialogue in COURIER font. COURIER closely resembles the screen display. Distinguishing the computer dialogue from the manual text is very important.

8.3.5 Date

Use the release date of the manual on the title page and footer.

8.3.6 Double Sided

Design documentation to be reproduced in a double-sided format. Any blank pages that are added for order of sections and organization of the manual should be numbered. No statement is necessary on blank pages.

8.3.7 Field Names

Use all uppercase letters (e.g., NAME).

8.3.8 File Names

Use initial capital letters (e.g., Patient file).

8.3.9 Fonts

8.3.9.1 Text

TIMES 10 POINT font is to be used throughout for the text, with TIMES 10 POINT BOLD used for headings and items to be emphasized, such as Notes, Cautions, or Warnings.

8.3.9.2 Computer Prompts

COURIER 10 POINT CAPS is to be used to indicate the computer prompts. Computer printouts and sample reports are to be so annotated.

8.3.9.3 User Response

HELVETICA 10 POINT BOLD will be used in the computer-interactive sections to indicate the information typed by the user in response to the computer prompts.

8.3.10 Headers and Footers

Headers and footers are required for all manuals. The information contained in the headers and footers is:

- Package name and version number
- Release date (month/year)

- Page number
- Manual type (User, Technical, etc.)
- Chapter or Section name
- Header and footer information can be arranged at the discretion of the documenter.
- Lines separating the header and footer from text is mandatory.

8.3.11 Headings

Place headings at the left margin (block style), distinguish them in bold or uppercase, and do not underline.

8.3.12 Margins

1.0-inch top and bottom with headers and footers placed in the margin area; left and right margins shall be 0.75 inch.

8.3.13 Option Names

Use initial capitalization to set off option names. Option and file names (since they both use initial caps) should be distinguished with the body of text by the word option or file. For example, Enter or Edit File Entries option and Patient file. Do not use single or double quotes with options.

8.3.14 Page Dimensions

8.5 by 11 inches.

8.3.15 Package Names

Use initial capitalization when referring to package names (e.g., Laboratory). Initially completely spell out the package name when used in text, after that time, the abbreviated package name may be used.

8.3.16 Page Numbers

Place page numbers in the footer. All pages should be numbered except for the title pages and their blank back sheet. Use lower case Roman numerals to number pages that contain the table of contents, preface, acknowledgements or lists of tables.

8.3.17 Paragraph Numbers

Paragraph numbers, if used, shall be in legal format throughout the document. The lowest level allowable in outline style shall be level four (e.g., paragraphs 1.1.1.1, 1.1.12, etc.)

8.3.18 Prompts

Use double quotes around prompts used within text (e.g., "Select PATIENT NAME:"). Use single quotes within double quotes only. Don’t use quotes around prompts in a recreated computer dialogue.

8.3.19 Return Key/Enter Key

Use this symbol: RETURN when referring to users entering information. The RETURN symbol does not need to be underlined. It is recommended that this key be defined in the orientation section of the manual.

8.3.20 Sub-Headings

Place sub-headings at left margin (block style), distinguish them in bold or uppercase, smaller size type than the heading, and do not underline.

8.3.21 Text

In most cases, the text is to be presented flush left with no paragraph indents and no hyphenation.

8.3.22 Up-Arrow

Use this symbol: ^ when referring to the up-arrow or hat. Do not use single quotes around the up-arrow or hat. It is recommended that this key be defined in the Orientation section of the manual.

8.3.23 Version Number

Spell out and capitalize (or initial cap) the word version when used in the cover (e.g., Laboratory Version 5.0). Abbreviate version to V space when used in the header or footer (e.g., Lab V 5.1).

8.3.24 Computer Menus/Options

In the manuals, computer menus or options are shown inside a box resembling a computer screen. Use a different type of box for report examples. Be consistent in whatever format is chosen and use it throughout the entire document.

8.3.25 Wording Convention

"Press" rather than "strike" or "hit" is used in instructions - i.e., "Press RETURN". Likewise, "Type" is preferred to "enter" in instructions to the user - i.e., "Type TRANSACTION".

9 APPENDIX G - VISTA/RPMS GUI STANDARDS

(Note: these standards are still pending VA approval and may need some modification for IHS)

1. Purpose The purpose of these GUI Standards is to provide a basic structure that can be applied to every software package, to provide consistency in all user interface software, and to provide criteria to follow for national certification.

2. VistA/RPMS GUI Standards All VHA Decentralized Hospital Computer Program (VistA)/RPMS user interface software will, at a minimum, meet the following standards and comply with the spirit of the guidelines (in the respective reference works).

2.1 Reference works For all user interfaces for applications that run in the Microsoft Windows graphical environment, the book, The Windows Interface, An Application Design Guide (for Macintosh, ISBN: 0-201-62216-5) and The Windows Interface Guidelines for Software Design (for Windows 95 and Windows NT, ISBN: 1-55615-679-0) shall be adhered to unless modified by this document. In addition, to make applications more accessible to individuals who have disabilities or who are aging, the document Designing Accessible Applications (draft version 15 or later) from Microsoft Corporation, should be followed.

For all user interfaces for applications that run in other graphical environments, corresponding guidelines should be followed. For example, for the Macintosh, the Macintosh Human Interface Guidelines shall be adhered to unless modified by this document.

Standard metaphors shall be used in the major application areas of clinical, management, and support applications. The standard metaphors shall be approved by the Application Requirements Group responsible for the application area.

3. VistA/RPMS GUI Guidelines

3.1 General Principles

The intent of using this standard in VistA/RPMS applications is to provide the end-user with enough consistency in the use of VistA/RPMS applications that he or she can approach applications with confidence, and readily transfer knowledge gained from previous experience to new applications and functions. For presentation options not yet covered by standards, developers must apply the following three criteria that have been and will continue to be the underpinnings of an effective interface.

3.1.1 Simplicity

Simplicity will be achieved through limiting each single operation to one that can achieve predictable closure, provide informative feedback to every action, allow easy reversal of actions, and generally ensure that each dialog is the result of action initiated by the user.

3.1.2 Consistency

By being consistent throughout all applications the VistA will maximize the opportunity for users to transfer and apply skills to new applications. Consistency will be achieved, to a great extent, through the widespread reuse of applications software. The use of standard controls and adherence to conventions widely used in commercial products are important components of consistency.

3.1.3 Intuitiveness

Intuitiveness (or self-evidency) will be achieved through the uniform use of the "Object/Action" paradigm, whereby the user selects an object (e.g., an icon, a file, or a block of text) and selects an action that is to be applied to that object (e.g., an icon, a file, or a block of text) and selects an action that is to be applied to the object (e.g., run, delete, or copy). Intuitiveness is also achieved through use of real world metaphors, where objects are identified in a way that is meaningful to the user. Intuitiveness will be supplemented with a rich help facility throughout all VistA/RPMS applications.

4. GUI Controls

At the user level, the human interface consists of controls or widgets that permit the user to manipulate the information presented there. Graphical User Interface (GUI) metaphors, for the most part, still control procedural applications. The GUI may be built from reusable objects and may portray the information presented as data objects, but the underlying

applications and their data are often procedural. Only when both the interface and the underlying applications are data-centric can the user interface itself be an object-oriented user interface (OOUI).

Since the manipulation of data in a GUI is still application-centered rather than document-data-centered, the GUI creates only an illusion of data objects. When that illusion is convincing, the difference between GUI and OOUI is insignificant. When it is not possible to maintain that illusion, the difference is painfully obvious.

For instance, using Cut-and-Paste editing or Drag-and-Drop manipulation, makes it easy to move a prescription to a progress note. The prescription information can then appear there as a text string. However, a progress note does not accept a prescription as a container accepts a piece of paper. Instead, it changes the representation of the prescription's structured information to create a text string. The user cannot readily invert the process by using Drag-and-Drop to move text string prescription from a progress note back to the medication profile.

In their current form, these guidelines define some components of VistA/RPMS GUIs. The intent is to provide a descriptive framework to help developers produce the same GUI functionality using different tools on different platforms. The framework is user-centered; the framework specifies what the user does and sees at the GUI rather than how the GUI or its underlying applications accomplish it.

These guides address general issues as well as details concerning the controls from which the interface is built. Menus, check boxes, buttons of various kinds are all controls - visual components of the GUI. They are characterized here from the users' viewpoint; they are seen to consist of Revealed Features, and to exhibit certain Standard Behaviors. The user perceives a control's Revealed Features visually. Its associated Standard Behaviors are perceived as responses of the control to various stimuli (mouse motions, clicks and keystrokes). This characterization, while similar to the object-oriented language paradigm of object properties and methods, does not correspond exactly to that model.

Most controls have several Revealed Features. A check box, for instance, consists of an area that indicates whether it is checked (ON or OFF) and an area that displays its caption. Some controls may exhibit different behaviors when the same stimulus is applied to different features of the same control.

In order to create an interface that users will be able to learn, every control must demonstrate predictable behavior. All controls that present the same Revealed Features must exhibit the same Standard Behaviors.

5. General Guidelines

5.1 General Guideline #1 Controls having the same Revealed Features must have the same Standard Behaviors. If a novel behavior is required of an existing control, then a new widget is required for that control. The new widget must have a unique visual

appearance to allow the user to distinguish it from existing widgets.

Windows and dialogs are controls too, and they should obey General Guideline #1. Application windows, modal dialogs, and semi-modal dialogs should each have a distinctive visual appearance. Each distinctive appearance should be associated with its own standard behaviors. For example all interactive warnings of a given severity should appear in dialogs having the same visual appearance, reveal a predictable choice of options for exiting from the dialog, and use the same icons.

The "natural" reading sequence in English is left to right from top to bottom. The organization of text and data entry controls in each window should respect this convention.

5.2 General Guideline #2a The preferred organization of controls within a window is one that is correctly interpreted by the user when read according to the natural style of the user's native written language. For English this is left-to-right across the page from top-to-bottom.

5.3 Guideline #2b When space limitations dictate an alternative presentation, a newspaper column type organization is permissible. The sequence in which controls are Focused when the user completes an action or the user strikes a special key indicating that the previous or next control should be focused (often accomplished with cursor keys or TAB key) should follow the user's natural reading style. The first portion of the screen presents static, or read-only information (an exception would be a time & date that is presented by the system but could be modified by the user). The central portion of the active area contains controls that can be selected and varied within the window, but are generally not affected until the window is closed or the dialog is exited. Command buttons that effect action are placed to be encountered after other information has been presented and the user has a basis for selecting some particular action or mix of actions.

For example, with columns of radio buttons or check boxes, the tab order would be from the top of the column to the bottom, then over to the next column. Likewise, such order would hold for panels in a window. There may also be exceptions where the center of attention would for example be in the center of the page, so the sequence might appropriately begin there.

5.4 General Guideline #3 The preferred sequencing of controls within a window is one that moves the focus according to General Guideline #2. Not every control within a window need to be enabled. Controls (menus and other widgets) should share a common appearance that indicates that they are inactive. In many GUI environments this appearance is a gray shade (the control or its caption appears dimmed) that informs the user that the control cannot be used. Controls that cannot be used should not accept the Focus; they should be skipped.

5.5 General Guideline #4a All disabled controls must have a similar, distinctive visual

appearance (ex: dimmed) and must not be allowed to accept the Focus.

Appropriate use of inactivation limits the user's possible actions at every point in the user session to those choices that are valid. This prevents many needless error messages and dialogs.

5.6 General Guideline #4b Any control (or a possible choice within a control) that is not valid at a given moment must appear disabled.

5.7 General Guideline #4c Although GUI controls are usually activated using a pointing device, it is important that users be able to perform all tasks using only their keyboards. This not only suits the preferences of some users but also provides for an alternative interface control method should the pointer be lost or broken. Because uniformity and simplicity are crucial to making the GUI easy to learn and memorable, there must be a uniform keyboard method for accessing GUI controls.

5.8 General Guideline #5 A control affording a keyboard shortcut or accelerator must identify the shortcut by emphasizing the appropriate character in its caption. The corresponding shortcut is then a Special key-Character keystroke combination. The choice of the Special key and the type of emphasis is determined by a convention consistent with the operating system and other applications.

In MS Windows, the preferred Special key is Alt and the type of emphasis is an underline. Thus, an underlined E means, "use the Alt-E keystroke combination." If a second Special key (ex: the Ctrl key) is used, it must have a corresponding, unique character emphasis. The preferred shortcut character is the first character of a control's caption (the first letter in a menu caption, for instance).

The choice of interface control captions (for menus, buttons, and other widgets) must involve user testing to select those captions that users most consistently identify with the action performed or option selected.

5.9 General Guideline #6 Control captions must be meaningful to the intended users.

Although the first character of a caption is the ideal shortcut character, user testing may reveal captions that are meaningful to users but when taken together create a conflict due to multiple occurrences of the same initial character. When this occurs, Guideline 7 must yield to Guideline 6.

5.10 General Guideline #7 The preferred shortcut key for a control is the first character of its caption.

Standard PC keyboards are equipped with a number of special function keys. When these are used (singly or combined with either Alt or Ctrl) to activate a control, the shortcut must appear as part of the control's caption so that it is visible. In MS Windows the convention is that the accelerator appears at the end of the caption (ex:

Spelling F7).

5.11 General Guideline #8 All shortcuts, whether implied (General Guideline #5) or explicit must be visible to the user.

Shortcuts for application-independent services that the operating system provides for application developers should follow operating system guidelines. Shortcuts for Cut, Paste, Copy, Print, and others occur in word processing, spreadsheet, and other non-VistA/RPMS applications with which the intended users may already be familiar. When these same actions are implemented in VistA/RPMS packages, the developers should conform to existing shortcut key conventions to provide inter-application consistency.

5.12 General Guideline #9 Application-independent shortcuts should conform to the target operating system's existing conventions.

Similarly, any task that is commonly accomplished in native applications by using operating system level common dialogs (ex: Font selection, palette color changes, etc.) should be handled in a new application with those same dialogs. The reason for this is that users change applications more frequently than they change platforms.

5.13 General Guideline #10 When a system level Common Dialog exists for an application function, it is preferable to use the system dialog rather than to create an application-specific dialog. At the user interface level, consistency among applications on a given platform trumps consistency of one application across platforms.

Similarly, the details for implementing direct manipulation techniques (drag-and-drop) should comply with the guidelines that apply to the host operating system and other applications on that platform. Again, consistency with other applications on the host platform trumps application consistency across platforms.

5.14 Windows-specific Guideline #11 With the mouse pointer on the source control, the user initiates Drag-and-drop by pressing the Left Mouse Button. With the mouse pointer over the destination control, the user completes the process by releasing the button. A change in the mouse pointer to an icon that represents the information contained in the source control signals the drag-initiation event. A change in the visual appearance of a control as the user drags a source over it signals that the underlying control may accept the source data (ex: a new border appears around the destination, the destination background color changes, etc.). Dragging over a control that cannot accept the source may be indicated by a change in the mouse pointer icon to the international NO symbol (a circle around a backslash).

6. Discussion

In general, anything that is standard for the GUI platform (Windows, Mac, etc.), should not be

controversial. Any control (or metaphor) used by any widely used commercial application (i.e., the top 3 or 4 word processors, spreadsheets, databases, etc.) is also not controversial. For example, the current Software Services client application development environment, Delphi/Visual Basic, provides a large number of components whose behaviors should not be superseded.

When deciding whether to create new object classes or to utilize existing components, if the added functionality of a newly designed class is not expected to be utilized by more than one object instance, that class should not be built. Instead, code should be within the appropriate event handler of an instance of an existing class.

Color - Color should never be the only means of setting apart information. In clinical settings, red may be reserved for "emergency" or "panic" use. In accounting, red may be a convention for a negative balance (which may be an "emergency" or "panic" circumstance for an accountant!).

Fonts - Fonts not exported with Windows should be avoided or be exported and installed with the application.

Content areas - Content areas that are "writeable" areas shall have a white background and content areas that are "read-only" shall have a pale yellow, slightly grayed, or other non-bright background.

Patient name selection - Delphi/Visual Basic components from the Software Services Standard Component Repository should be used for patient name selection.

Sound - Use of sound should be curtailed to rare and special circumstances.

These guidelines are minimal development constraints meant to help users develop a reliable "intuition" about GUI behavior. They are not a formula for producing a successful GUI. Following these guidelines at best ensures that VistA/RPMS GUIs will show some consistency. It does not guarantee that those GUIs will be usable. It is usability testing that will ultimately reveal both GUI design errors and missing VistA/RPMS functionality.